Student Name: **Student id:** **Sect#:** **Serial #:**

QUESTION #	1	2	3	4	5	TOTAL
MAX POINTS	10	10	14	14	15	
POINTS EARNED						

University of Bahrain College of Information Technology

Department of Computer Science

ITCS332: Concepts of Programming Languages **SECOND TEST** **Date: DEC 12, 2013**

****

QUESTION ONE: **[5+5 pts]**

- What will be printed after executing the C++ following code?

```
void funx(int t)
{ static int x1 = 3; x1 = x1 + t;
  int f = 2; f = f + x1 * t;
  cout << x1 << '\t' << f << endl;
  x1--;
}

int main()
{   int x = 5;
    x--;
    funx(x);
    x--;
    funx(x);
```

```
7       30
6       29
```

- **Consider the following Ada-like code.** The value of n printed by **print(n)** :

```
Procedure main is
  n: integer ;
  Procedure sub1 is
  begin
    n = n + 2 ; print (n) ;
  end ;
   Procedure sub2 is
     n : integer;
   begin
      n = 6 ; sub1 ; n = n + 5;
   end ;
begin
   n = 13 ; sub2 ;
end ;
```

1) Under static -scoped rules is 13+2= 15

2) Under dynamic-scoped rules is 6+2= 8

QUESTION TWO: **[5+5 pts]**

i. Write a predicate(s) named **"sum"** that takes a list of integer numbers named **"org"** , compares each pair of numbers in **"org"** , and calculates the sum of the larger elements in each pair. **If** the list **"org"** contains one value, then the sum equals to this value. (See examples below).

```
?- sum([],N).
N = 0.

?- sum([17],N).
N = 17 .

?- sum([5,10],N).
N = 10.

?- sum([5,4,8,3],N).
N = 13 .

?- sum([5,4,8,3,10],N).
N = 23 .
```

```
sum([],0).
sum([SL],SL).
sum([H1,H2|T], SL) :- H1>=H2, HR is H1, sum(T, TR), SL is HR + TR.
sum([H1,H2|T], SL) :- H1<H2, HR is H2, sum(T, TR), SL is HR + TR.
```

ii. Write Prolog predicate(s) named **"test"** that takes a list of integer numbers named **"org"** and produces another list named **"new"**. The elements of **"new"**are calculated as follows: (See examples below).

- If the element in the list **"org"** is **even,** then the next element in **"new"** equals to the half of the element.
- If the element in the list **"org"** is **odd,** then the next element in **"new"** equals to the square of the element.

```
?- test([5],X).
X = [25]

?- test([5,6,11],U).
U = [25, 3, 121]

?- test([9,5,7,10,19,6,8,23],T).
T = [81, 25, 49, 5, 361, 3, 4, 529]
```

```
test([],[]).
test([H|T],[HR|TR]):- H1 is H mod 2,H1 =:= 0,HR is H //2,test(T,TR).
test([H|T],[HR|TR]):- H1 is H mod 2,H1 =\= 0,HR is H*H,test(T,TR).
```

QUESTION THREE: **[14 pts]**

- List 2 differences between stack-dynamic and explicit-heap dynamic variables. **[2x2 pts]**

  1) **Stack-dynamic variables** are allocated on the stack, while **explicit-heap dynamic** are allocated from the heap.

  2) **Stack-dynamic variables** are allocated/deallocated automatically, while **explicit-heap dynamic** are allocated/deallocated by special instructions of the programmer.

- Name the three types of C++ arrays and provide ONE code example for each array type. **[3x2 pts]**

  3) Static array.

     static int x[100].

  4) Fixed-stack dynamic array.

     void f1(...)

     { int t[50]; ... }

  5) Fixed-heap dynamic array.

     int *ptrX = new int x[100].

- Study the following C_like code and answer the four questions below: **[4x1 pts]**

```
static int z[40];
void FUNCX (int size)
{  int w[40];      int y[size];
   int x[]={9,5,11,-23,17,29};
   int *f = new int[40];
   ...
   delete [] f;
}
void char FUNCY ()
{    char *uptr = new char('U');
     static float d = 3.75;
     double sum; ...
}
```

  6) The lifetime of array **z** begin when **the program is loaded** and ends when **the program terminates.**

  7) The scope of a variable **sum** is **from its declaration statement in the function funcy** until **the last statement of the function funcy.**

  8) The lifetime of a variable **d** begins when **the function funcy is called first time** and ends when **the entire program terminates.**

  9) The lifetime of a variable **sum** begins whenever **the function funcy is called** and ends when **the function funcy terminates (return).**

QUESTION FOUR: **MCQ Questions** **[14 pts]**

1) Given the C++ declaration: "`double G[60];`" the address of the element `G[25] = base +`
   a) `24*8` b) `25*8` c) `25*4` d) `60*8` e) None
2) The subprogram call is bound to its code in the library at _____ time.
   a) Compilation b) Loading c) Language Implementation d) Language design e) Linking
3) For every assignment to a subrange variable, types are checked for compatibility at _____ time.
   a) Compile b) Load c) Run d) Language design e) Link
4) A character string whose length is specified when it is declared is called _____ string.
   a) Static length b) Dynamic length c) Limited dynamic length d) Circular length
5) The code to access an array element is generated by the _____.
   a) Loader b) Linker c) Programmer d) Compiler e) None
6) The variable attribute that specifies where the variable can be accessed in a program is:
   a) Type b) Lifetime c) Scope d) Address e) None
7) The variable attribute defined as a time period between allocation and de-allocation of a variable is:
   a) Type b) Lifetime c) Scope d) Address e) None
8) One of the following types is NOT ordinal:
   a) bool b) enumeration c) char d) double e) int
9) The types of the subscripts allowed in array references are decided by:
   a) Programmer b) Language designer c) Linker d) Language implementer e) None
10) For multidimensional arrays, row-major or column-major ordering is selected by:
   a) Compiler b) Language designer c) Linker d) Language implementer e) None
11) In C++, the arrays allocated/de-allocated using new/delete operators are of type _____
   a) Explicit-heap dynamic b) Heap dynamic c) Fixed stack dynamic d) None
12) One of the following is NOT determined by the data type of an object
   a) Operations b) Values Range c) Scope d) Precision
13) The array that can grow and shrink during the execution time is of type _____.
   a) Fixed heap dynamic b) Heap dynamic c) Fixed stack dynamic d) Static
14) For every assignment to a subrange variable, range checking is done at _____ time.
   a) Compile b) Load c) Run d) Language design e) Link

Question #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Answer	B	E	A	A	D	C	B	D	B	D	D	C	B	C

QUESTION FIVE: ***Fill in blanks Questions*** **[15 pts]**

1) The dynamic length strings are implemented using Linked Lists or Adjacent memory Cells.
2) In currently used programming languages character string are defined as Primary or Structured
3) In languages with dynamic type binding, the type may be specified by executing assignment statement and Type inferencing.
4) Name 2 character string design issues: String type: primitive or structured and String length: static or dynamic.
5) The index type in any array reference is specified by Language Designer. The storage size of a given data type is specified by Compiler Constructor.
6) The 2 major disadvantages of static variables are: They do not support recursion and Low efficient use of memory space.
7) The 2 disadvantages of dynamic length string implemented as a linked list are: complex and slow string operations and pointer space overhead.
8) A data type is a collection of data values (objects) plus a set of predefined operations that can be applied on those objects.
9) In JAVA, aliases are created using references or methods parameters.
10) The main advantage of long names is improved readability and its main disadvantage is increased symbol table size.
11) The two kinds of compatibility methods are: name compatibility and structure compatibility
12) In C++, the static scope is created for every block or function.
13) The initial value of a statically bound to storage variable must be a literal/constant expression; while the initial value of a dynamically bound to storage variable can be any expression.
14) The major advantage of stack-dynamic variables is: efficient use of memory space and its major disadvantage is time overhead for allocation and deallocation.
15) Dynamic scope of variables is based on the Calling sequence of program units enclosing those variables. While static scope is based on the spatial textual layout of program units/blocks.